DOI: 10.1111/cgf.13938
EUROGRAPHICS 2020 / U. Assarsson and D. Panozzo
(Guest Editors)

Volume 39 (2020), Number 2

Fast and Robust Stochastic Structural Optimization

Qiaodong Cui,[1] Timothy Langlois,[2] Pradeep Sen,[1] Theodore Kim[3]

University of California, Santa Barbara[1] Adobe Research[2] Yale University[3]

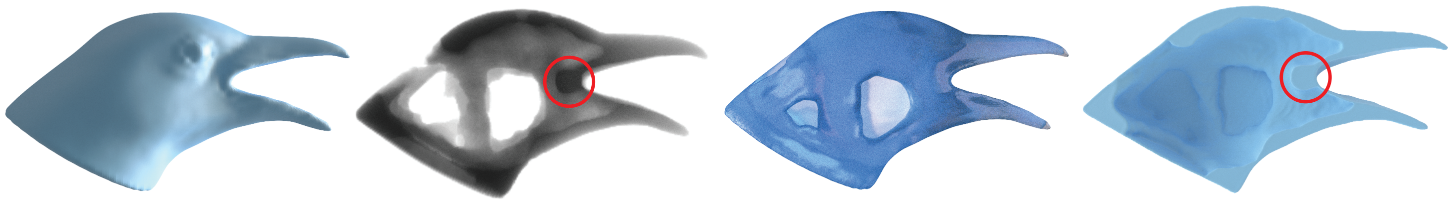


Figure 1: *A raven model optimized using our method. From left to right: The initial object, the optimized density field, the final shape, and a cut-away view of the final shape. Our method automatically reinforces the corner (commissure) of the beak, circled in red, which is likely to break under real-world impacts. Material is subtracted from regions that are unlikely to experience impacts.*

Abstract

Stochastic structural analysis can assess whether a fabricated object will break under real-world conditions. While this approach is powerful, it is also quite slow, which has previously limited its use to coarse resolutions (e.g., $26 \times 34 \times 28$). We show that this approach can be made asymptotically faster, which in practice reduces computation time by two orders of magnitude, and allows the use of previously-infeasible resolutions. We achieve this by showing that the probability gradient can be computed in linear time instead of quadratic, and by using a robust new scheme that stabilizes the inertia gradients used by the optimization. Additionally, we propose a constrained restart method that deals with local minima, and a sheathing approach that further reduces the weight of the shape. Together, these components enable the discovery of previously-inaccessible designs.

1. Introduction

Recently, a wide range of computational techniques have been developed to assist in the design of objects manufactured using additive fabrication (see, e.g., excellent surveys of the state-of-the-art [LEM*17, MS19]). One key concern is the robustness of the manufactured object (i.e., the conditions in which it will break), and various *failure analysis* algorithms have been developed to estimate the failure cases of the object in order to improve its design.

Traditionally, failure analysis is performed for worst-case scenarios, and while such analyses are critical in certain scenarios (e.g., bridge construction), they can be overly conservative in others, such as figurine design. In these (more common) situations, it is more realistic to ask whether an object is robust in real-world situations, such as being dropped on the floor or down a flight of stairs. In this case, reinforcing portions of the object that are unlikely to experience an impact because they are already shielded by more robust regions is clearly sub-optimal. Unlikely mechanical scenarios should not artificially constrain the design space, and the manufactured designs should be optimized for the "common case."

Along these lines, Langlois et al. [LSD*16] recently proposed a

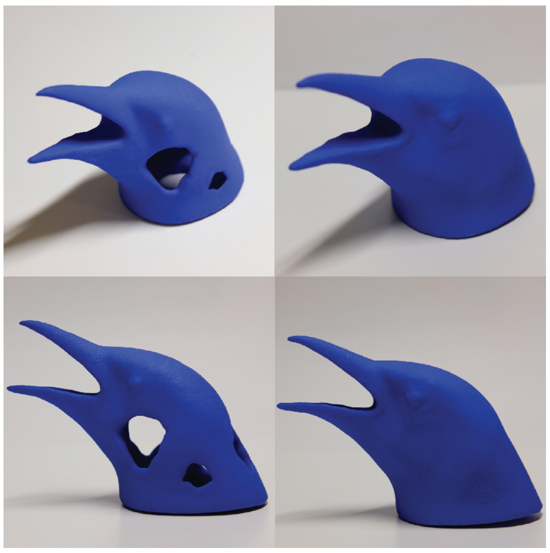


Figure 2: *Photos of the 3D printed results from the raven in Fig. 1.*

stochastic structural analysis method that used a rigid body simulator to sample a more realistic space of real-world impacts and construct a probability map of predicted failures. This probability map was then incorporated as a constraint into a topology optimization, and used to solve a context-aware inverse design problem that produced geometrical models more robust to realistic impacts. Unfortunately, the method is quite computationally intensive, where even *a single optimization step* over a small $26 \times 34 \times 28$ model takes over an *hour* to compute .

We observe that this computational expense arises for two main reasons. First, computing the probability gradients is quadratic in the number of elements. Second, the inertia gradient that arises from the rigid body simulation is computed using a finite difference scheme that leads to instabilities, which in turn negatively impacts the convergence of the optimization. In this paper, we present an approach that is both asymptotically faster and more numerically robust than Langlois et al. [LSD*16]. We achieve both of these goals directly; no approximation is applied to the original algorithm. In the first case, we show that a careful analysis of the probability gradient can yield a computation that is only *linear* in the number of elements. In the second case, we use a combination of Gaussian Mixture Models and an alternate central-differencing method to arrive at a more stable version of the inertia gradient.

Even with these improvements, the optimization can still stall at local minima. We therefore introduce a *constrained restart* method that is able to make further progress and discover interesting new structures that were otherwise inaccessible to the original algorithm. Finally, we show that additional progress can be made by assuming that a thin sheath that is beyond the resolution of the numerical simulation will be printed along the exterior of the object. Our individual technical contributions are as follows:

- Asymptotically faster computation of probability gradients
- A robust scheme for computing inertia gradients that stabilizes the optimization search direction
- A constrained restart strategy that allows the global optimization to climb out of local minima
- A sheathing approach that robustly removes additional mass from the final design

Together, these components comprise a stochastic structural optimization method that is orders of magnitude faster than Langlois et al. [LSD*16], able to achieve previously-impractical resolutions, and capable of discovering previously-inaccessible designs.

2. Related Work

Since structural optimization was introduced [Ben89, BS95], it has attracted lots of attention in computer graphics. Traditionally, topology optimization involves minimization of compliance. There are many works in computer graphics following this approach, e.g., [WWY*13, LHZ*18, LSZ*14]. However, compliance minimization can overfit to one loading condition, and has difficulty predicting whether objects will fail under realistic conditions.

Instead of minimizing compliance, minimizing the object's weight subject to a stress constraint offers a better guarantee of the robustness of the object. For example, Lee et al. [LJM12] uses a constraint on the yield stress and minimizes the weight of the object. Similarly, Ulu et al. [UMK19] optimizes the thickness of the shell to minimize the weight, subject to a constraint on the yield stress. Many of these methods use prescribed loadings. This can be useful in some particular cases, e.g., designing a bridge, where the loading can be prescribed in advance. However, in many other cases, the loading might be unknown beforehand.

To capture uncertainty, stochastic finite element analysis has been extensively studied in the engineering community. Stefanou [Ste09] provides an excellent overview. There are two broad classes of methods: 1) the perturbation approach, which uses a Taylor series expansion of the system matrix and solution [LBM86], and 2) the spectral stochastic finite element method, which represents each solution quantity with a series of random Hermite polynomials [GS91]. Monte Carlo simulation (MCS) [PP96] can be used in conjunction with these two methods, which models randomness by solving a deterministic problem many times using different samples of the random variables. These methods are very general, and can consider uncertainties in the loading, geometry, and material behavior of the problem. In our case, we care only about uncertainty in loading conditions, so do not need the complex, expensive machinery to approximate randomness in the system matrix provided by these methods. Our approach (and the previous approach we accelerate [LSD*16]) is akin to an MCS approach using model reduction to reduce the computational load.

To address uncertainty in the graphics community, worst case structural analysis was introduced by Zhou et al. [ZPZ13]. The method computes a worst case loading scenario where it produces the worst possible stress distribution in the object. Along this line, several different works use the worst case loading to optimize the structure [UMK17, PRZ17, SZB18]. However, it is unknown how often the worst cast loading will be present in a realistic scenario, such as a figurine falling and hitting the ground.

To address this limitation, Langlois et al. [LSD*16] presented a method where the loading of the object was computed from a rigid body simulator that closely mimicked realistic loadings. The work also introduced semantically meaningful failure probabilities that better reflected real-word object failures. The work also presented a structural optimization scheme where the weight of the object was minimized under failure probability constraints. Still, this optimization scheme remained expensive due to the computation of the failure probability gradients.

Our work immediately follows this line. We aim to optimize stochastic structural optimization through three specific contributions: acceleration of the gradient computation, a more robust probability gradient formulation, and a restart strategy to overcome non-optimal local minima during optimization.

3. Stochastic Structural Optimization

Here we briefly summarize the stochastic structural optimization technique of Langlois et al. [LSD*16].

Notation: We use unbolded lower case for scalars, bolded lower case for vectors ($\boldsymbol{\omega}$), and bolded upper case for matrices ($\mathbf{K}$). When an indexing operation results in a scalar, we unbold the result. We use a colon to denote double-contraction between matrices.

Finite Element Method (FEM): We use a hexahedral uniform grid as our FEM discretization and compute the displacement ($\mathbf{u}$) and element Cauchy stresses ($\boldsymbol{\sigma}$) that arise from an external force ($\mathbf{f}$):

$$\begin{aligned}\mathbf{Ku} &= \mathbf{f}\\ \boldsymbol{\sigma} &= \mathbf{CBu}.\end{aligned} \tag{1}$$

Above, $\mathbf{K} \in \mathbb{R}^{3n\times 3n}$ is the stiffness matrix, $\boldsymbol{\sigma} \in \mathbb{R}^{6m}$ is a vector of per-element Cauchy stresses, $\mathbf{C} \in \mathbb{R}^{6m\times 6m}$ is a constitutive matrix, and $\mathbf{B} \in \mathbb{R}^{6m\times 3n}$ maps $\mathbf{u}$ to Cauchy strains. Stresses and strains are evaluated at the center of each voxel. The quantity m is the total number of mesh elements, and n is the number of vertices.

Using the per-element stress,

$$\boldsymbol{\sigma}_e = \begin{bmatrix}\sigma_{11} & \sigma_{12} & \sigma_{31}\\ \sigma_{12} & \sigma_{22} & \sigma_{23}\\ \sigma_{31} & \sigma_{23} & \sigma_{33}\end{bmatrix} \tag{2}$$

we compute a scalar, per-element von Mises stress:

$$\begin{aligned}S(\boldsymbol{\sigma}_e) =& \frac{1}{2}\left[(\sigma_{11}-\sigma_{22})^2 + (\sigma_{22}-\sigma_{33})^2 + (\sigma_{33}-\sigma_{11})^2\right] + \\ & 3\left(\sigma_{23}^2 + \sigma_{31}^2 + \sigma_{12}^2\right).\end{aligned} \tag{3}$$

A yield stress $\hat{\sigma}$ completes our failure criteria. An element fails if $S(\boldsymbol{\sigma}_e) > \hat{\sigma}$, and we define "object failure" as the failure of any individual element.

Stochastic Failure Probability: Prior to optimization, we need to estimate an object's failure probability under various force distributions. Langlois et al. [LSD*16] estimated real-world loadings by using a rigid-body simulation to generate force samples. For each sample, they then computed the maximal von Mises stress across the whole object, and then estimated a probability distribution function (PDF) of maximal stresses. The object fails if its maximal stress is greater than the yield stress, so the survival probability is computed by integrating the PDF from 0 to $\hat{\sigma}$. The failure probability is one minus the survival probability.

Many force samples (≈ 5000) are needed to accurately represent the PDF, so principal component analysis (PCA) is used to reduce its computational complexity. We denote each force sample as $\mathbf{f}^i$, where $i = 1\dots n_s$, and n_s is the total number of samples. In lieu of performing an FEM analysis for each sample, they compute a reduced force basis, $\bar{\mathbf{F}} \in \mathbb{R}^{3n\times r}$, where r is the number of principal components. Each sample is then represented with a reduced coordinate, $\boldsymbol{\alpha}^i \in \mathbb{R}^r$, where $\mathbf{f}^i \approx \bar{\mathbf{F}}\boldsymbol{\alpha}^i$. The element stresses for each sample i can then be computed as:

$$\boldsymbol{\sigma}^i = \mathbf{CBK}^{-1}\bar{\mathbf{F}}\boldsymbol{\alpha}^i. \tag{4}$$

By pre-solving $\mathbf{CBK}^{-1}$ for each column in $\bar{\mathbf{F}}$, significant savings can be achieved when $r \ll n_s$ (e.g. $r \approx 100$).

The normalized, whole-object stress for each $\mathbf{f}^i$ is then:

$$s^i = \frac{1}{\hat{\sigma}}\max_e(S(\boldsymbol{\sigma}_e^i)) \begin{cases} e = 1\dots m\\ i = 1\dots n_s\end{cases} \tag{5}$$

Using all the stress samples s^i, we can construct a probability distribution function (PDF) $p(s)$ for the whole-object stress. The corresponding cumulative distribution (CDF) function is denoted as $P(s)$. The probability of the object survival $P(s < 1)$ is then:

$$P(s<1) = \int_0^1 p(s)ds. \tag{6}$$

Topology Optimization: The failure probability is then used as a constraint in a topology optimization that adds or subtracts materials from some initial design. The overall goal is to reduce the final object weight while satisfying a user defined failure probability Θ:

$$\begin{aligned}&\min \sum_{e=1}^{m} \omega_e\\ &\text{s.t.}\;\; P(s<1) > 1-\Theta.\end{aligned} \tag{7}$$

Above, $\boldsymbol{\omega}$ is a vector of element densities, such that $\forall e\ \omega_e \in [0,1]$, which is usually initialized to be fully filled ($\forall e,\ \omega_e = 1$). Eqn. 7 is optimized using the Method of Moving Asymptotes (MMA) [Sva02], which requires both the object and constraint gradients. The objective gradient is straightforward to efficiently compute, but the constraint gradient is a major bottleneck because the existing approach is quadratic in the number of elements. We will show that it is possible to reduce its complexity to linear.

4. Our Method

First, in §4.1, we analyze the complexity of computing the probability gradients. By carefully leveraging the structure of the problem, we found the *quadratic* complexity of previous methods [LSD*16] can be reduced to *linear* with respect to the number of elements. In practice, this results in a roughly two-order of magnitude speedup.

Next, in §4.2, we show that the existing approach leads to unreliable probability gradients. This negatively impacts the convergence of the optimization and, in turn, the final design. We show how to stabilize these gradients, which leads to higher-quality shapes.

The optimization can still stall at local minima. To address this, we introduce (§4.3) a constrained restart method that identifies and constrains promising structures when the optimization stalls, and applies a perturbation to bump the state out of its local minimum. Finally, we make additional progress by allowing the outer shell of the object to erode, and later restore the visual appearance of the object by adding a lightweight sheath as a post-process.

4.1. Asymptotic Analysis and Acceleration

First, we will analyze the existing method [LSD*16] to show that it runs in $O(m^2)$. Then we will show that an identical computation can be done in $O(m)$.

4.1.1. The Previous Quadratic Method

We begin by expanding the derivative of $p(s)$ from Eqn. 6 in terms of s^i using the chain rule:

$$\frac{\partial P(s<1)}{\partial \boldsymbol{\omega}} = \int_0^1 \sum_{i=1}^{n_s} \frac{\partial p}{\partial s^i}\frac{\partial s^i}{\partial \boldsymbol{\omega}} ds. \tag{8}$$

The $\frac{\partial s^i}{\partial \boldsymbol{\omega}}$ term is computed by replacing the discontinuous max function in Eqn. 5 with a smoother L^p norm. The density term ω_e is also multiplied beforehand to avoid singularities as shown in [LJM12], $s^i \approx ||\sqrt{\omega_e^i} S(\boldsymbol{\sigma}_e^i)||_p / \hat{\sigma}$, yielding:

$$\frac{\partial s^i}{\partial \boldsymbol{\omega}} = \frac{1}{\hat{\sigma}} \left(\sum_{e=1}^{m} \sqrt{\omega_e^i} S(\boldsymbol{\sigma}_e^i)^p \right)^{\frac{1}{p}-1} \sum_{e=1}^{m} (\sqrt{\omega_e^i} S(\boldsymbol{\sigma}_e^i))^{p-1} \left(\frac{1}{2\sqrt{\omega_e^i}} S(\boldsymbol{\sigma}_e^i) + \frac{\partial S}{\partial \boldsymbol{\sigma}_e^i} \frac{\partial \boldsymbol{\sigma}_e^i}{\partial \boldsymbol{\omega}} \right) \tag{9}$$

Combining equation 9 with equation 8, we obtain:

$$\frac{\partial P(s<1)}{\partial \boldsymbol{\omega}} = \sum_{i=1}^{n_s} a^i \left(\mathbf{b}^i + \left(\frac{\partial \boldsymbol{\sigma}^i}{\partial \boldsymbol{\omega}} \right)^T \mathbf{c}^i \right) \tag{10}$$

where

$$a^i = \int_0^1 \frac{\partial p}{\partial s^i} ds \frac{1}{\hat{\sigma}} \left(\sum_{e=1}^{m} \sqrt{\omega_e^i} S(\boldsymbol{\sigma}_e^i)^p \right)^{\frac{1}{p}-1}$$
$$\mathbf{b}^i = \sum_{e=1}^{m} (\sqrt{\omega_e^i} S(\boldsymbol{\sigma}_e^i))^{p-1} S(\boldsymbol{\sigma}_e^i)/(2\sqrt{\omega_e^i})$$
$$\mathbf{c}_e^i = (\sqrt{\omega_e^i} S(\boldsymbol{\sigma}_e^i))^{p-1} \frac{\partial S}{\partial \boldsymbol{\sigma}_e^i}$$

$\frac{\partial S}{\partial \boldsymbol{\sigma}_e^i}$ can be computed from equation 3. Next, from equation 4 we compute:

$$\frac{\partial \boldsymbol{\sigma}^i}{\partial \boldsymbol{\omega}} = \underbrace{-\mathbf{CBK}^{-1} \frac{\partial \mathbf{K}}{\partial \boldsymbol{\omega}} \bar{\mathbf{U}} \boldsymbol{\alpha}^i}_{\mathbf{I}} + \underbrace{\mathbf{CBK}^{-1} \frac{\partial \bar{\mathbf{F}}}{\partial \boldsymbol{\omega}} \boldsymbol{\alpha}^i}_{\mathbf{II}} + \underbrace{\mathbf{CB}\bar{\mathbf{U}} \frac{\partial \boldsymbol{\alpha}^i}{\partial \boldsymbol{\omega}}}_{\mathbf{III}} \tag{11}$$

where $\bar{\mathbf{U}} = \mathbf{K}^{-1}\bar{\mathbf{F}}$. Here, **I** computes the derivative of the stiffness matrix, **II** computes the gradient of the reduced force basis, and **III** computes the gradient of the reduced force coordinates.

Combining equations 10 and 11, we obtain the final probability derivative, which can be computed as:

$$\frac{\partial P(s<1)}{\partial \boldsymbol{\omega}} = (\mathbf{K}^{-1}\mathbf{Y}\bar{\mathbf{U}}^T) : \frac{\partial \mathbf{K}}{\partial \boldsymbol{\omega}} + (\mathbf{K}^{-1}\mathbf{Y}) : \frac{\partial \bar{\mathbf{F}}}{\partial \boldsymbol{\omega}} + \mathbf{x} + \mathbf{t} \tag{12}$$

where

$$\mathbf{Y} = \sum_{i=1}^{n_s} \mathbf{B}^T \mathbf{C}^T \mathbf{c}^i \otimes \boldsymbol{\alpha}^i \qquad \mathbf{t} = \sum_{i=1}^{n_s} a^i \mathbf{b}^i$$
$$\mathbf{x} = \sum_{i=1}^{n_s} \frac{\partial \boldsymbol{\alpha}^i}{\partial \boldsymbol{\omega}} \bar{\mathbf{U}}^T \mathbf{B}^T \mathbf{C}^T \mathbf{c}^i.$$

The main complexity in Eqn. 12 lies in the second term, $(\mathbf{K}^{-1}\mathbf{Y}) : \frac{\partial \bar{\mathbf{F}}}{\partial \boldsymbol{\omega}}$. We show in Appendix A for a single element e, the force derivative on the right of the double-contraction can be written as

$$\frac{\partial \bar{\mathbf{F}}}{\partial \boldsymbol{\omega}_e} = \bar{\mathbf{F}} \mathbf{W}_e. \tag{13}$$

$\mathbf{W}_e \in \mathbb{R}^{r \times r}$ is a dense matrix that has to be evaluated for *each* element. A naïve evaluation for one element then becomes mr^2 because $\bar{\mathbf{F}} \in \mathbb{R}^{3n \times r}$ and $m \propto n$ due to the uniform grid discretization. Computing the force derivatives for all samples is then $O(m^2 r^2)$.

4.1.2. Our Linear Method

We first observe that in equation 13, the per-element force basis matrix $\bar{\mathbf{F}}$ is fixed, and only the smaller $\mathbf{W}_e \in \mathbb{R}^{r \times r}$ ever changes. Second, we observe that the final quantity $(\mathbf{K}^{-1}\mathbf{Y}) : \frac{\partial \bar{\mathbf{F}}}{\partial \boldsymbol{\omega}}$ is all that matters; the per-element intermediate $\frac{\partial \bar{\mathbf{F}}}{\partial \boldsymbol{\omega}_e}$ is not strictly required. Therefore, if we pre-contract $\bar{\mathbf{F}}$ with $\mathbf{K}^{-1}\mathbf{Y}$, we obtain a smaller matrix, $\bar{\mathbf{F}}^T \mathbf{K}^{-1} \mathbf{Y} \in \mathbb{R}^{r \times r}$. Each element can then be computed as $\bar{\mathbf{F}}^T \mathbf{K}^{-1} \mathbf{Y} : \mathbf{W}_e$, which is only $O(r^2)$ per element.

Specifically, each element must compute the product

$$g_e = \mathbf{K}^{-1}\mathbf{Y} : (\bar{\mathbf{F}} \mathbf{W}_e), \tag{14}$$

where g_e is the entry of the second term in Eqn. 12 for element e. Both $\mathbf{Z} = \mathbf{K}^{-1}\mathbf{Y}$ and $\bar{\mathbf{F}}$ are static, per-element, $\mathbb{R}^{3n \times r}$ matrices that we use to rewrite g_e as

$$g_e = \sum_{i,j} (\mathbf{Z})_{ij} (\bar{\mathbf{F}} \mathbf{W}_e)_{ij} = \sum_{i,j,k} \mathbf{Z}_{ij} \bar{\mathbf{F}}_{ik} \mathbf{W}_{kje} = \sum_{i,j,k} \bar{\mathbf{F}}_{ik} \mathbf{Z}_{ij} \mathbf{W}_{kje} \\ = (\bar{\mathbf{F}}^T \mathbf{Z}) : \mathbf{W}_e. \tag{15}$$

Since $\bar{\mathbf{F}}^T \mathbf{Z} \in \mathbb{R}^{r \times r}$ is fixed for all e, we can precompute it in $O(mr^2)$ time. At runtime, an $O(r^2)$ contraction is performed over m elements, yielding an $O(mr^2)$ overall running time.

So far, we have only examined the derivative of the force basis in Eqn. 13. However, naïvely evaluating $\mathbf{W}_e$ also takes $O(m^2)$ time. We show in Appendix B that this can also be reduced to $O(m)$ by leveraging matrix sparsity and similar pre-computations.

We show in Table 1 the running time of computing the total probability gradients using the previous quadratic method (estimated) and our linear method. Our method is asymptotically faster, and accelerates this stage of the method by two orders of magnitude, effectively removing it as the bottleneck of the method.

	Time (s)		
Resolution	[LSD*16]	**Our Method**	**Speedup**
$28 \times 32 \times 36$	$1.84^\dagger$	0.041	**44.9×**
$28 \times 44 \times 28$	$6.18^\dagger$	0.044	**140×**
$40 \times 64 \times 60$	$41.7^\dagger$	0.128	**326×**

Table 1: *Running time of Eqn. 13 using the quadratic method of Langlois et al. [LSD*16] and our linear method.* †*Estimated time.*

4.2. Stabilizing the Inertia Gradients

4.2.1. Previous Method

We examine the relevant term from Eqn. 12,

$$\mathbf{x} = \sum_{i=1}^{n_s} \frac{\partial \boldsymbol{\alpha}^i}{\partial \boldsymbol{\omega}} \bar{\mathbf{U}}^T \mathbf{B}^T \mathbf{C}^T \mathbf{c}^i, \tag{16}$$

that measures how changing the voxel densities $\boldsymbol{\omega}$ influence the rigid body force samples $\boldsymbol{\alpha}^i$. The previous method [LSD*16] evaluates this term using finite differences.

Given n_s samples $\boldsymbol{\alpha}^i \in \mathbb{R}^r$, we assume each entry of $\boldsymbol{\alpha}^i$ is sampled from a 1D PDF, and take its finite difference over that distribution. The j^{th} entry of each force sample α_j is denoted α. We assume α is a random variable and that α_j^i is drawn from the PDF $c(\alpha)$.

Instead of computing a finite difference for the sample α_j^i, we perform a finite difference over its distribution. For any α, we can compute its CDF as $C(\alpha)$, and retrieve α_j^i by sampling its inverse using $\alpha_j^i = C^{-1}(u)$, where u is a uniform random variable. Using the properties of the CDF, we obtain:

$$\begin{aligned}\frac{\partial C(C^{-1}(u))}{\partial \boldsymbol{\omega}} &= \frac{\partial u}{\partial \boldsymbol{\omega}} = 0\\ \frac{\partial C(C^{-1}(u))}{\partial \boldsymbol{\omega}} &= \left.\frac{\partial C}{\partial \alpha}\right|_{\alpha_j^i} \left.\frac{\partial C^{-1}}{\partial \boldsymbol{\omega}}\right|_u + \left.\frac{\partial C}{\partial \boldsymbol{\omega}}\right|_\alpha = 0.\end{aligned} \tag{17}$$

By manipulating Eqn. 17, we obtain the derivative of α:

$$\frac{\partial \alpha}{\partial \boldsymbol{\omega}} = \left.\frac{\partial C^{-1}}{\partial \boldsymbol{\omega}}\right|_u = -\frac{1}{c(\alpha)} \left.\frac{\partial C}{\partial \boldsymbol{\omega}}\right|_\alpha. \tag{18}$$

We insert $\alpha = \alpha_j^i$ into Eqn. 18 to compute the gradient for each random sample. Therefore, an important step in computing the inertia gradients is building the distributions $C(\alpha)$ and $c(\alpha)$ over the samples α_j^i, where $i = 1 \dots n_s$.

Langlois et al. [LSD*16] computed $C(\alpha)$ and $c(\alpha)$ by fitting a uniform 1D finite element grid over the samples, which represents them as a sum of basis functions: $c(\alpha) = \sum_{i=1}^k a_i \psi_i(\alpha)$. Here, k is the number of elements of this 1D finite element grid, a_i is a shape coefficient, and $\psi_i(\alpha)$ is a symmetric hat function of element i. A Galerkin method is then used to solve for a_i. However, this approach can fit the data poorly. In the top of Fig. 3, we show the PDF $c(\alpha)$ that results from this approach using n elements. For a small n, the histogram is fit poorly. As n increases, ringing artifacts appear. This leads to instabilities when computing $\frac{1}{c(\alpha)}$ in Eqn. 18 because the ringing artifacts can cause the PDF to become negative.

4.2.2. Stabilized Inertia Gradients

We use Gaussian Mixture Models (GMMs) to address this problem. GMMs are widely used to capture discrete distributions [Bis06], and we found that they yield results superior to the FEM approach.

We first expand $c(\alpha)$ using a set of Gaussians:

$$c(\alpha) = \sum_{i=1}^k \pi_i \mathcal{N}(\alpha \,|\, \mu_k, \sigma_k) \tag{19}$$

where k is the number of Gaussians, π_k is a weight, and $\mathcal{N}(\alpha \,|\, \mu_k, \sigma_k)$ is a Gaussian with mean μ_k and variance σ_k. The parameters, π_k, σ_k, μ_k, can be computed from α_j^i using expectation maximization [Bis06], and we found that $k = 10$ usually yields good results. This leads to the superior distribution representations we show in the bottom of Fig. 3.

Returning to Eqn. 18, $\frac{\partial C}{\partial \boldsymbol{\omega}}$ is also evaluated using finite differences. We assume that all the potential contact positions of the rigid-body lies on the surface of the object. So the rigid-body can be parameterized by its mass, center of mass, and moment of inertia. As a result, α_j^i and its CDF $C(\alpha)$ can be parameterized using

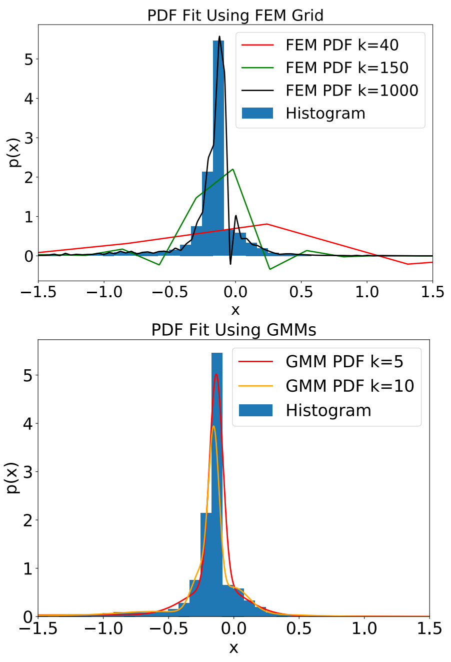


Figure 3: *The PDF constructed from an FEM grid fits the data poorly and leads to unstable probability gradients. The distribution has a long tail; we have zoomed into a portion of the x axis. Our GMM fit does not exhibit these artifacts.*

a total of 10 variables. Denoting the parameters as $M_i, i = 1 \dots 10$, the finite difference is computed as:

$$\left.\frac{\partial C}{\partial \boldsymbol{\omega}}\right|_\alpha = \sum_{i=1}^{10} \left.\frac{\partial C}{\partial M_i}\right|_\alpha \frac{\partial M_i}{\partial \boldsymbol{\omega}}. \tag{20}$$

The $\frac{\partial M_i}{\partial \boldsymbol{\omega}}$ is straightforward, and we use a centered difference for $\frac{\partial C}{\partial M_i}$:

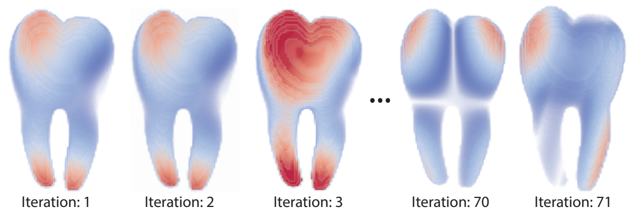
$$\left.\frac{\partial C}{\partial M_i}\right|_\alpha = \frac{1}{2} \frac{C(M_i + \Delta M)|_\alpha - C(M_i - \Delta M)|_\alpha}{\Delta M}. \tag{21}$$

Evaluating the above equation involves two extra rounds of rigid body simulation with $M_i + \Delta M$ and $M_i - \Delta M$ perturbations applied to each parameter, for a total of 20 rounds.

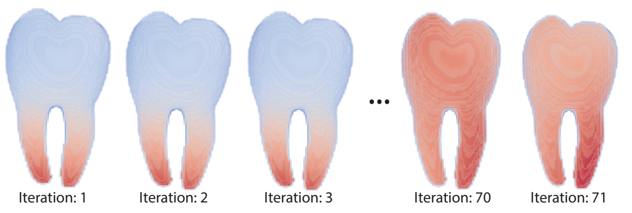
The resulting probability distribution leads to much stabler inertia gradients, as shown in Fig. 4. In that figure, we show the probability gradients from the first three frames of the optimization as well as two frames near the end. The element densities change only slightly during these frames, and yet the FEM-based gradients oscillate wildly. Using our method, the gradient becomes very stable, and greatly improves the convergence of the optimization (Fig. 12).

4.3. Constrained Restart Strategy

Eqn. 7 has a non-linear constraint and a large number of variables, so we use the Method of Moving Asymptotes (MMA) [Sva02] as our optimization algorithm. The non-linearity of the probability constraint causes the optimization to often fall out of the feasible



(a) *Previous, unstable probability gradients*



(b) *Our stable probability gradients*

Figure 4: *Probability gradients from the first three optimization frames and two frames near the end. The previous method changes wildly, even when the underlying densities change very little. Our method changes slowly, in step with the density changes.*

region. MMA can recover, but often at the cost of oscillatory behavior that converges to sub-optimal local minima [Sva02]. One popular technique for addressing this well-known problem is the Solid Isotropic Material with Penalization (SIMP) model [BS95], but we found it to be insufficient for our case (see Appendix D).

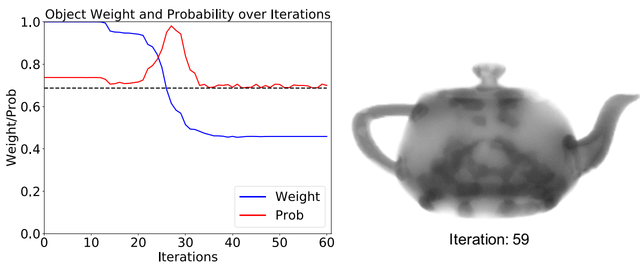


Figure 5: *Left: The object weight and survival probability during the optimization. Right: The local optimum at iteration 59.*

Instead, we found the following *constrained restart* method to be effective, which can be viewed as a form of either block-cyclic reduction [GVL13] or sand-filling [BMS15]. We observe that when the optimization stalls, it usually has found a preliminary, but promising, reinforcement structure. Therefore, we perform multiple optimization passes where the promising structures from the previous iteration are used as an initial guess.

We isolate these structures by applying threshold c to the current solution and constraining the results to $\omega_e = 1$ in Eqn. 7. Next, we add a perturbation to push the global solution state out of its current local minimum. We compute extension density field $\boldsymbol{\beta}$ of the stalled solution $\boldsymbol{\omega}^{i-1}$ and then use $\min(\boldsymbol{\omega}^{i-1} + \boldsymbol{\beta}, 1)$ as the initial state for the next pass. The extension field is computed as:

$$\boldsymbol{\beta} = \max\left(1 - \frac{\mathrm{SDF}(V)}{b_\beta}, 0\right) \tag{22}$$

where SDF is a signed distance field, V is the set of constrained voxels and b_β is a bandwidth parameter. Fig. 6 illustrates these quantities. The strategy essentially inflates the existing reinforcement structure for the next optimization pass.

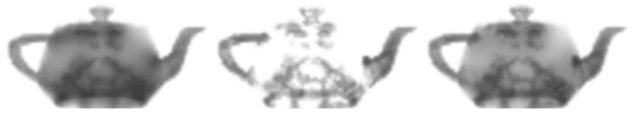


Figure 6: *Left: Solution $\boldsymbol{\omega}^{i-1}$ after one optimization pass. Middle: The reinforcement structure V from $\boldsymbol{\omega}^{i-1}$. Right: Extension field $\boldsymbol{\beta}$ of V.*

Using the extension field $\boldsymbol{\beta}$ and constraints on V, we run the next optimization pass. We found this to be effective in perturbing the solution from local minima and finding sparser and more interesting structures, e.g. as shown in Fig. 7. The complete optimization is listed in Algorithm 1. We found that $n_{opt} = 3$ rounds of optimization with MMA produces converged results. As shown in Fig. 8, $n_{opt} > 3$ produces little change. In all our computations we set $c = 0.7$ and $b_\beta = 16$ grid cells.

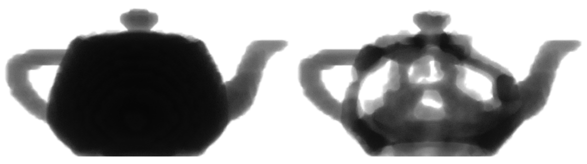


Figure 7: *Left: Initial state $\min(\boldsymbol{\omega}^{i-1} + \boldsymbol{\beta}, 1)$ for the next round of optimization. Right: Converged result $\boldsymbol{\omega}^{i}$ after the next round.*

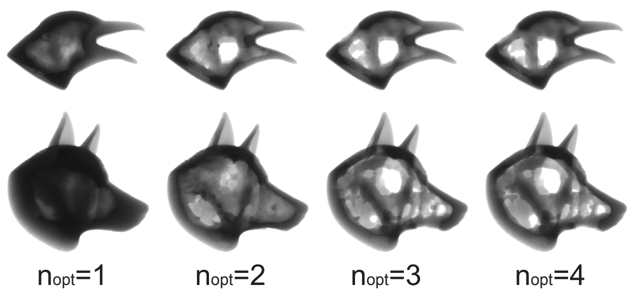


Figure 8: *Results with different n_{opt}. We set $n_{opt} = 3$ because $n_{opt} > 3$ yielded negligible improvements.*

Algorithm 1 Incremental Shape Optimization

```
Input: User-defined geometry , user defined failure probability $\Theta$
1: procedure SHAPE OPTIMIZATION
2:     $\boldsymbol{\omega}^1 \leftarrow \mathbf{1}$
3:     Optimize $\boldsymbol{\omega}^1$ using MMA to convergence
4:     $V \leftarrow$ voxels in $\boldsymbol{\omega}^1$ with density larger than $c$
5:     for $i$ from 2 to $n_{opt}$ do
6:         Compute $\boldsymbol{\beta}$ from $V$
7:         $\boldsymbol{\omega}^i \leftarrow \min(\boldsymbol{\omega}^{i-1} + \boldsymbol{\beta}, 1)$
8:         Constrain densities of $V$ to 1
9:         Optimize $\boldsymbol{\omega}^i$ using MMA to convergence
10:        $V \leftarrow$ voxels in $\boldsymbol{\omega}^i$ with density larger than $c$
11:    end for
12:    $V \leftarrow \emptyset$
13:    $\boldsymbol{\omega}^f \leftarrow \boldsymbol{\omega}^m$
14:    Optimize $\boldsymbol{\omega}^f$ using MMA to convergence
15: end procedure
```

4.4. Sheathing Post-Process

Many algorithms constrain the exterior of the object during the optimization [UMK17, LSD*16]. However, the final object then has an outer shell that is the thickness of the (quite coarse) voxel grid. However, as the shell is thickened, it can become the main reinforcement structure in the model [UMK17], which biases the optimization towards shell-like designs, and results in heavier objects. Instead, we only constrain the shell at force contact locations, allowing the optimizer to find a lighter reinforcement structure, and allowing surface regions which are unlikely to experience contact to be hollowed out. To preserve the surface geometry, we perform a post-process that adds a thin "sheath" of material, far below the resolution that we can simulate, along the original surface. We show in §5.4 that this sheathing has a minimal impact on the object's performance.

5. Implementation and Results

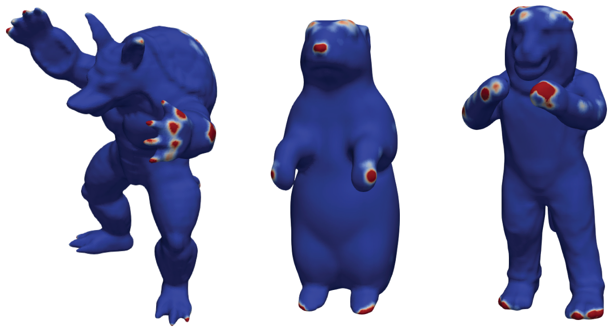


Figure 9: *Force contact locations for different examples. Surfaces marked red are possible contact locations.*

5.1. Implementation Details

We initially voxelize the surface mesh according to the resolutions in Tbl. 4. We use Bullet [C*13] to obtain rigid body force samples. For all our examples, we use the following scenario: the shape falls 1 meter with a random initial orientation and small random angular velocity, and hits a flat plane. We record the three initial contact events when the shape hits the ground. We use $n_s = 5000$ rigid body simulations in all our examples and kept 90% of the total variance when performing PCA on the force samples. To avoid checkerboard patterns, we augment our cost function with the energy term of from Schumacher et al. [SBR*15]. We constrain the rigid body contact locations to $\omega_e = 1$ to ensure that they do not change during optimization. As shown in Fig 9, these locations are usually very sparse and lie on the object's convex hull. We use the following material parameters: Young's modulus = 2.2 GPa, density = 1.037 g/cm^3, and yield stress = 0.031 GPa. The object is scaled so that the maximal bounding box dimension is 15 cm.

We implemented our algorithm in C++. OpenMP multi-threading was used whenever possible, Eigen [GJ*10] is used for most matrix operations, Intel's Paradiso was used to solve linear systems, and Armadillo [SC16] was used for the GMMs. All our results were run on a desktop with 192 GB of memory and a 2.4 GHz, 20-core Intel 6148.

5.2. Optimization Results

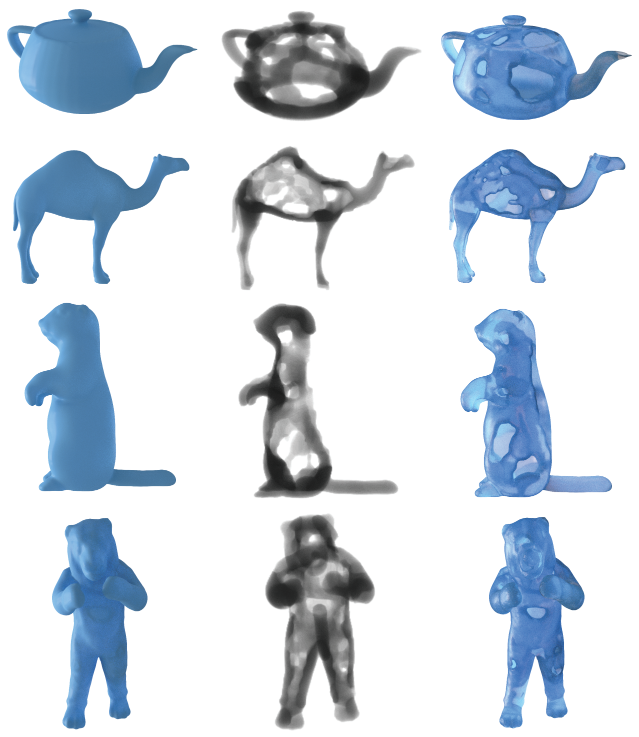


Figure 10: *Left: The teapot, camel, beaver, and tiger surface meshes. Middle: Final optimized density field. Right: Final meshed results, rendered translucently to show internal structure.*

Fig. 10 shows that our method adds densities to locations of potential high stress. In particular, fragile locations such as the head of the beaver, the extremities of the camel, the handle and spout tip of the teapot, the beak corner (commissure) on the crow (Fig. 1), and the pelvis of the Armadillo (Fig. 15, top) are reinforced during optimization. Conversely, material is removed from low-stress regions. We do not constrain the surface, so interesting contact-dependent structures form, such as the removal of the beaver's back, the tiger's chest, and the divot between the camel's humps.

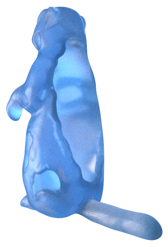
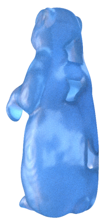



Figure 11: *Left: Optimized result of a beaver with its tail. Collision with its back becomes very unlikely, so the region is entirely hollowed out. Right: With the tail removed, the back experiences collisions, so the material remains.*

To further explore the dependence of our algorithm on real-world contacts, we optimized a beaver model with and without its tail (Fig. 11). The optimizer aggressively subtracts material along the beaver's back when collisions in that region become unlikely. Fig. 12 compares our results to Langlois et al. [LSD*16], and shows that our method consistently produces a superior (lighter) object. Tbl. 2 shows the survival probability remains similar, and we achieve up to 3.22× lighter objects. Fig. 13 shows the object weights and survival probabilities during optimization. Langlois et al. [LSD*16] stalls early, while our method makes steady progress.

Model	Method	Weight (gram)	Improvement	Survival Probability
Rabbit	[LSD*16]	97.3		0.630
	Ours	36.7	**2.65×**	0.662
Penguin	[LSD*16]	94.8		0.650
	Ours	29.4	**3.22×**	0.767
Molar	[LSD*16]	63.6		0.598
	Ours	32.3	**1.96×**	0.621

Table 2: *The final object weight and survival probabilities of our method and Langlois et al. [LSD*16]. Our method consistently produces lighter objects with similar survival probabilities.*

5.3. Post-Processing

As is common in many algorithms, we optimize over continuous densities (Fig. 14, left). To obtain the final mesh (Fig. 14, middle), we use marching cubes on the density field [LC87] to obtain the $\omega = 0.5$ isocontour. Since we did not constrain the exterior shell of the object, we attach a thin shell of width $\frac{dx}{4}$ using the SDF of the original mesh, where dx is length of one hexahedron (Fig. 14, right). We show in §5.4 that this thin shell has a negligible effect on the final object's weight and survival probability.

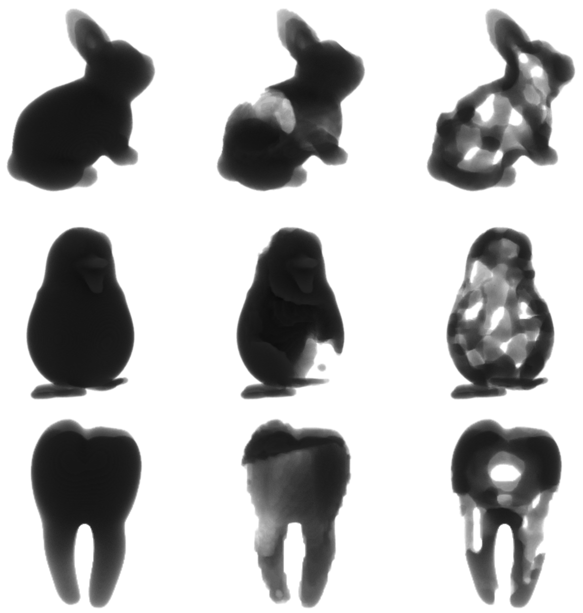


Figure 12: *The Langlois et al. [LSD*16] algorithm vs. ours. Left: Original shape. Middle: Stalled result from Langlois et al. Right: Improved result using ours.*

5.4. Optimization Validation

We ran several comparisons to validate our sheathing post-process. First, we compared the results of the optimization with and without shell constraints (Tbl. 3). The shell constraint consistently produces heavier results, even through the survival probability is almost identical. We then added a thin sheath to the results without the shell constraint. The survival probability remains essentially the same (in the Armadillo case, it actually *improves*), and the object weight remains significantly below that found using the shell constraint.

Model	Configuration	Weight (gram)	Survival Probability
Armadillo	with shell constraint	86	0.706
	w/o shell constraint	56	0.706
	sheathing post-process	66	0.700
Raven	with shell constraint	63	0.624
	w/o shell constraint	42	0.624
	sheathing post-process	48	0.632
Teapot	with shell constraint	75	0.737
	w/o shell constraint	52	0.737
	sheathing post-process	62	0.747
Tiger	with shell constraint	65	0.813
	w/o shell constraint	45	0.813
	sheathing post-process	52	0.819

Table 3: *Results with and without the shell constraint, and with the sheathing post-process from §4.4.*

Additionally, we visualized the von Mises stresses for several models in Fig. 15. The heavier shell-constrained models produce

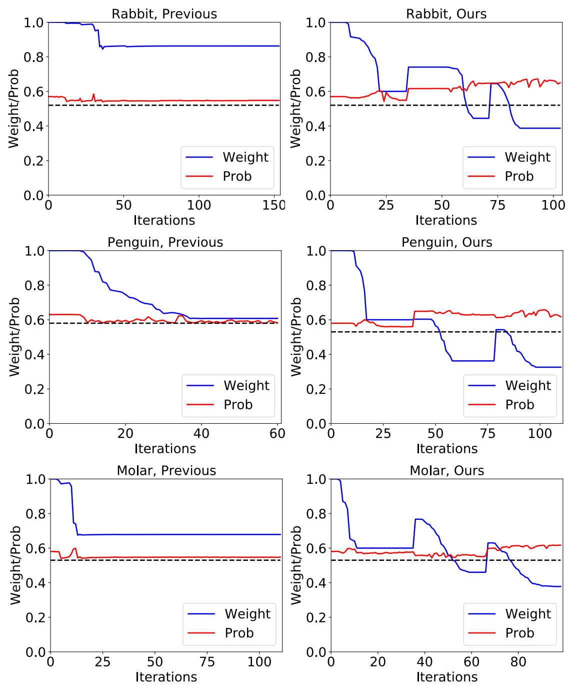


Figure 13: *Left: The object weights and survival probabilities when optimizing using Langlois et al. [LSD*16]. The convergence consistently stalls. Right: The same plots using our method. Top to bottom: The rabbit, penguin, and molar models. The dashed line is the constraint probability. In our method, the curves jump at optimization restarts, but eventually reach lower weights.*

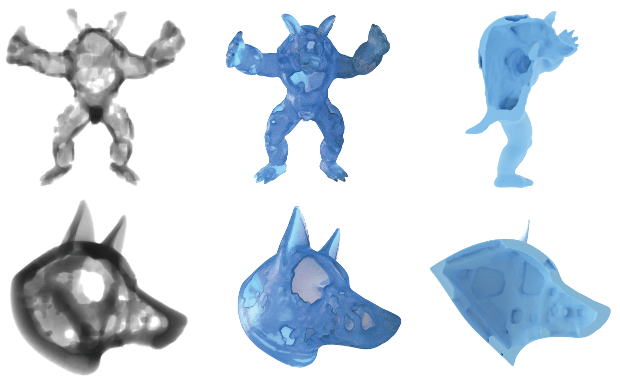


Figure 14: *Left: Optimized density field. Middle: Post processed final shapes. Right: Cut view of the post processed shapes.*

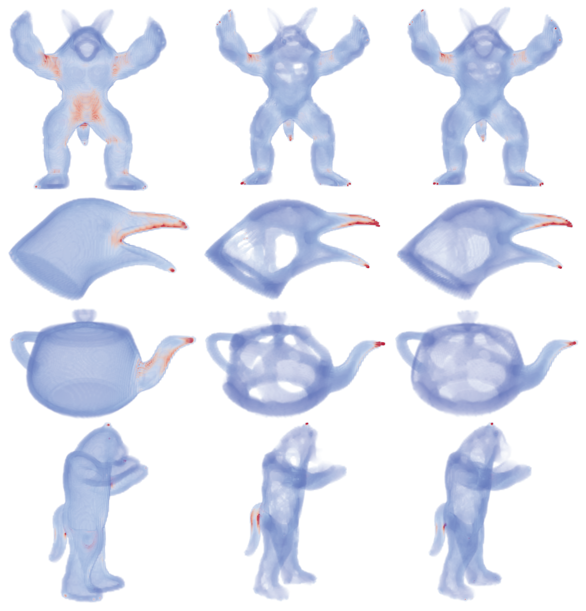


Figure 15: *Left: The von Mises stress of models with a shell constraint. Middle: Stresses without a shell constraint. Right: Stresses after a sheath is added. The stresses appear in essentially the same regions. Notice the optimization with a shell constraint produces a higher stress in the pelvis of the armadillo.*

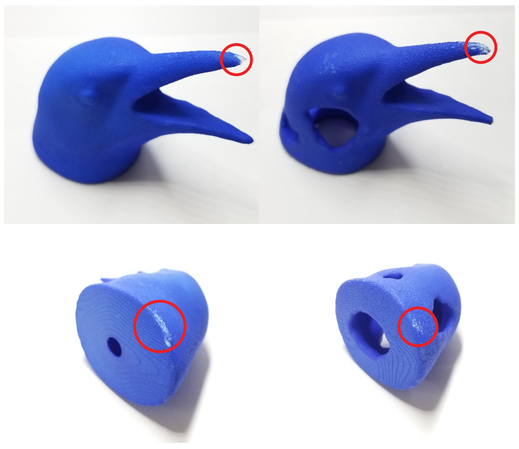


Figure 16: *Photographs of breakage patterns. Left: Breakages of the un-sheathed object. Right: Breakages of the sheathed object.*

larger rigid body impact forces, and therefore larger stresses. The sheath post-processed models have almost the same stress distribution as the original shell-unconstrained results, which indicate that the regions of likely impact have been effectively reinforced.

6. Physical Validation

We printed five copies of our Raven model with the sheath and five copies without. The results are shown in Fig. 2.

We then dropped each model from a height of 1.5 meters until a breakage occurred. Each drop used a random initial orientation, and the results are listed in Tbl. 5. We computed the expected survival probabilities from these breakage statistics, and found that they are close to the probability predicted by our simulation.

In Fig. 16 we show two breakage patterns from the sheathed and un-sheathed objects. Breaks occur around the beak or contact locations. In all five of our tests, as predicted by our simulation, the sheath did not come into contact with the ground.

Model	**Resolution**	**# Iters**	**Time (s)** Sampling	GMMs	FEM Solve	Grad	Total	**Volume Reduction**	**Survival Probability**
Armadillo	$56 \times 64 \times 52$	189	72.52	31.40	8.36	289.17	401.45	67.4%	0.706
Camel	$20 \times 64 \times 52$	142	70.14	20.00	3.71	140.34	234.19	64.8%	0.734
Beaver(tail)	$56 \times 28 \times 64$	162	60.55	22.56	6.28	200.84	290.23	70.4%	0.617
Raven	$32 \times 64 \times 40$	159	62.11	31.00	5.00	260.70	328.18	73.1%	0.624
Teapot	$64 \times 36 \times 44$	168	60.80	43.00	7.51	282.51	393.83	71.7%	0.737
Dog	$40 \times 64 \times 60$	103	74.69	32.44	11.8	341.81	460.71	77.4%	0.754
Tiger	$32 \times 40 \times 64$	134	67.36	31.04	4.93	170.77	274.10	67.1%	0.813
Penguin	$36 \times 48 \times 40$	97	21.74	42.11	11.5	180.51	255.85	71.3%	0.767
Rabbit	$28 \times 44 \times 48$	103	17.28	35.15	6.57	99.72	158.72	68.9%	0.662
Hand	$40 \times 32 \times 24$	98	30.72	28.28	3.80	57.66	120.46	68.8%	0.868
Molar	$32 \times 28 \times 48$	91	33.41	41.87	5.45	127.03	207.76	60.0%	0.621
Panda	$28 \times 32 \times 36$	95	17.22	35.12	4.54	90.30	147.18	68.2%	0.776

Table 4: *Timing breakdown (in seconds), volume reduction, and survival probability across different examples. The volume reduction is computed with final after the sheathing post-process.*

sheathed		un-sheathed	
weight	# of drops	weight	# of drops
44.02 g	4	41.68 g	5
45.00 g	4	42.62 g	3
44.84 g	3	42.48 g	3
44.44 g	3	41.77 g	2
44.89 g	2	41.26 g	2

Table 5: *Number of experimental drops of the sheathed and un-sheathed objects before breakage. Our predicted probability of remaining intact is respectively 0.632 and 0.624. The expected probability of remaining intact computed from the breakage statistics and assuming a binomial distribution is respectively 0.688 and 0.666.*

7. Discussion and Future Work

We have described an asymptotically faster and more robust method for stochastic structural optimization. We reduced the previous quadric complexity to linear, which results in a two order of magnitude speed-up. By stabilizing the gradients and utilizing a constrained restart method, we achieve better convergence.

We hold the contact points fixed during the optimization, which enables us to use a finite difference method to compute the inertia gradients (i.e. the gradient of the rigid body simulator). However, this means that our method cannot handle topological changes along the surface, so it limited to examples where the external surface is prescribed. As a direction for future work, allowing the gradients to incorporate shape changes would broaden the possible application areas.

While the constrained restart method works in practice, it remains to be seen if the reinforcement structure can be identified and constrained within a single optimization pass to improve convergence. Even with our improvements, computing the probability gradient can still be a bottleneck due to its dense linear algebra operations, which limits the resolution of our method. One way to reduce this cost could be to use a sparse grid [LHZ*18].

References

[Ben89] Bendsøe M. P.: Optimal shape design as a material distribution problem. *Structural optimization 1*, 4 (1989), 193–202. 2

[Bis06] Bishop C. M.: *Pattern recognition and machine learning*. springer, 2006. 5

[BMS15] Bernardi R. C., Melo M. C., Schulten K.: Enhanced sampling techniques in molecular dynamics simulations of biological systems. *Biochimica et Biophysica Acta (BBA)-General Subjects 1850*, 5 (2015), 872–877. 6

[BS95] Bendsøe M. P., Sigmund O.: *Optimization of structural topology, shape, and material*, vol. 414. Springer, 1995. 2, 6

[C*13] Coumans E., et al.: Bullet physics library. *Open source: bulletphysics. org 15*, 49 (2013), 5. 7

[GJ*10] Guennebaud G., Jacob B., et al.: Eigen v3. http://eigen.tuxfamily.org, 2010. 7

[GS91] Ghanem R. G., Spanos P. D.: *Stochastic Finite Elements: A Spectral Approach*. Springer-Verlag, 1991. 2

[GVL13] Golub G. H., Van Loan C. F.: Matrix computations. *The Johns Hopkins University Press* (2013). 6

[LBM86] Liu W. K., Belytschko T., Mani A.: Probabilistic finite elements for nonlinear structural dynamics. *Computer Methods in Applied Mechanics and Engineering 56*, 1 (1986), 61 – 81. 2

[LC87] Lorensen W. E., Cline H. E.: Marching cubes: A high resolution 3d surface construction algorithm. In *ACM SIGGRAPH computer graphics* (1987), vol. 21, ACM, pp. 163–169. 8

[LEM*17] Livesu M., Ellero S., Martínez J., Lefebvre S., Attene M.: From 3d models to 3d prints: an overview of the processing pipeline. In *Computer Graphics Forum* (2017), vol. 36, Wiley Online Library, pp. 537–564. 1

[LHZ*18] Liu H., Hu Y., Zhu B., Matusik W., Sifakis E.: Narrow-band topology optimization on a sparsely populated grid. *ACM Trans. Graph. 37*, 6 (Dec. 2018), 251:1–251:14. 2, 10

[LJM12] Lee E., James K. A., Martins J. R.: Stress-constrained topology optimization with design-dependent loading. *Structural and Multidisciplinary Optimization 46*, 5 (2012), 647–661. 2, 4

[LSD*16] Langlois T., Shamir A., Dror D., Matusik W., Levin D. I. W.: Stochastic structural analysis for context-aware design and fabrication. *ACM Trans. Graph. 35*, 6 (Nov. 2016), 226:1–226:13. 1, 2, 3, 4, 5, 7, 8, 9, 11

[LSZ*14] Lu L., Sharf A., Zhao H., Wei Y., Fan Q., Chen X., Savoye Y., Tu C., Cohen-Or D., Chen B.: Build-to-last: Strength to weight 3d printed objects. *ACM Trans. Graph. 33*, 4 (July 2014), 97:1–97:10. 2

[MS19] Matusik W., Schulz A.: Computational fabrication. In *ACM SIGGRAPH Courses* (2019), pp. 7:1–7:305. 1

[PP96] Papadrakakis M., Papadopoulos V.: Robust and efficient methods for stochastic finite element analysis using monte carlo simulation. *Computer Methods in Applied Mechanics and Engineering 134*, 3 (1996), 325 – 340. 2

[PRZ17] Panetta J., Rahimian A., Zorin D.: Worst-case stress relief for microstructures. *ACM Trans. Graph. 36*, 4 (July 2017), 122:1–122:16. 2

[SBR*15] Schumacher C., Bickel B., Rys J., Marschner S., Daraio C., Gross M.: Microstructures to control elasticity in 3d printing. *ACM Transactions on Graphics (TOG) 34*, 4 (2015), 1–13. 7

[SC16] Sanderson C., Curtin R.: Armadillo: a template-based c++ library for linear algebra. *Journal of Open Source Software 1*, 2 (2016), 26. 7

[Ste09] Stefanou G.: The stochastic finite element method: Past, present and future. *Computer Methods in Applied Mechanics and Engineering 198*, 9 (2009), 1031 – 1051. 2

[Sva02] Svanberg K.: A class of globally convergent optimization methods based on conservative convex separable approximations. *SIAM Journal on Optimization 12*, 2 (2002), 555–573. 3, 5, 6

[SZB18] Schumacher C., Zehnder J., Bächer M.: Set-in-stone: Worst-case optimization of structures weak in tension. *ACM Trans. Graph. 37*, 6 (Dec. 2018), 252:1–252:13. 2

[UMK17] Ulu E., McCann J., Kara L. B.: Lightweight structure design under force location uncertainty. *ACM Trans. Graph. 36*, 4 (July 2017), 158:1–158:13. 2, 7

[UMK19] Ulu E., McCann J., Kara L. B.: Structural design using laplacian shells. *arXiv preprint arXiv:1906.10669* (2019). 2

[WWY*13] Wang W., Wang T. Y., Yang Z., Liu L., Tong X., Tong W., Deng J., Chen F., Liu X.: Cost-effective printing of 3d objects with skin-frame structures. *ACM Trans. Graph. 32*, 6 (Nov. 2013), 177:1–177:10. 2

[XSZB15] Xu H., Sin F., Zhu Y., Barbič J.: Nonlinear material design using principal stretches. *ACM Trans. Graph. 34*, 4 (July 2015), 75:1–75:11. 11

[ZPZ13] Zhou Q., Panetta J., Zorin D.: Worst-case structural analysis. *ACM Trans. Graph. 32*, 4 (July 2013), 137:1–137:12. 2

Appendix A: Evaluation of the Derivative of Reduced Force Vectors

For completeness, we summarize the derivative of the reduced force basis vectors $\bar{\mathbf{F}}$, which is described in the supplemental material of [LSD*16]. A finite element force sample $\mathbf{f}^i \in \mathbb{R}^{3n}$ is obtained by mapping a rigid body force $\mathbf{l}^i$ via a projection matrix J: $\mathbf{f}^i = \mathbf{J}\mathbf{l}^i$. Each rigid body force sample $\mathbf{l}^i \in \mathbb{R}^{3n_u+6}$ is of the form:

$$\mathbf{l}^i = \begin{bmatrix} \mathbf{l}_1 & \dots & \mathbf{l}_{n_u} & \mathbf{f}^{com} & \boldsymbol{\tau}^{com} \end{bmatrix} \tag{23}$$

where $\mathbf{l}_{3(j-1)} \in \mathbb{R}^3, j = 1, \dots, n_u$ are the contact forces sampled at the surface of the object, n_u is the total number of possible contact positions, and $\mathbf{f}^{com}, \boldsymbol{\tau}^{com} \in \mathbb{R}^3$ are the inertial force and torque acting on the center of the mass.

The matrix $\mathbf{J}$ is of the form:

$$\mathbf{J} = \begin{bmatrix} \mathbf{J}^m & \mathbf{J}^{com} & \mathbf{J}^{\tau} \end{bmatrix} \tag{24}$$

where $\mathbf{J}^m \in \mathbb{R}^{3n \times 3n_u}$ maps the surface contact points to volumetric element vertices, and $\mathbf{J}^{com}, \mathbf{J}^{\tau} \in \mathbb{R}^{3n \times 3}$ map the inertial forces and torques acting on the center of mass to element vertices.

Given n_s rigid body force samples, $\mathbf{L} = \begin{bmatrix} \mathbf{l}^1 & \dots & \mathbf{l}^{n_s} \end{bmatrix}$, the reduced force basis $\bar{\mathbf{F}}$ is the eigenvectors of the covariance matrix of the force samples, $\mathbf{JL}$. Therefore, we have $\mathbf{JLL}^T\mathbf{J}^T \approx \bar{\mathbf{F}}\boldsymbol{\Lambda}\bar{\mathbf{F}}^T$, where $\boldsymbol{\Lambda} \in \mathbb{R}^{r \times r}$ is a diagonal matrix of eigenvalues of the covariance matrix $\mathbf{JLL}^T\mathbf{J}^T$. From here we follow the procedure of [XSZB15]. We have:

$$\frac{\partial \mathbf{JLL}^T\mathbf{J}^T}{\partial \omega_e} = \frac{\partial \bar{\mathbf{F}}}{\partial \omega_e}\boldsymbol{\Lambda}\bar{\mathbf{F}}^T + \bar{\mathbf{F}}\frac{\partial \boldsymbol{\Lambda}}{\partial \omega_e}\bar{\mathbf{F}}^T + \bar{\mathbf{F}}\boldsymbol{\Lambda}\frac{\partial \bar{\mathbf{F}}^T}{\partial \omega_e} \tag{25}$$

Multiplying Eqn. 25 with $\bar{\mathbf{F}}^T$ from the left and $\bar{\mathbf{F}}$ from the right yields:

$$\bar{\mathbf{F}}^T\frac{\partial \mathbf{JLL}^T\mathbf{J}^T}{\partial \omega_e}\bar{\mathbf{F}} = \bar{\mathbf{F}}^T\frac{\partial \bar{\mathbf{F}}}{\partial \omega_e}\boldsymbol{\Lambda} + \frac{\partial \boldsymbol{\Lambda}}{\partial \omega_e} + \boldsymbol{\Lambda}\frac{\partial \bar{\mathbf{F}}^T}{\partial \omega_e}\bar{\mathbf{F}} \tag{26}$$

Denote $\mathbf{B}_e = \bar{\mathbf{F}}^T\frac{\partial \mathbf{JLL}^T\mathbf{J}^T}{\partial \omega_e}\bar{\mathbf{F}}$. The matrix $\mathbf{W}_e = \bar{\mathbf{F}}^T\frac{\partial \bar{\mathbf{F}}}{\partial \omega_e}$ is antisymmetric [XSZB15], therefore $\frac{\partial \boldsymbol{\Lambda}}{\partial \omega_e} = diag(\mathbf{B}_e)$, so we have:

$$\mathbf{B}_e - diag(\mathbf{B}_e) = \mathbf{W}_e\boldsymbol{\Lambda} + \boldsymbol{\Lambda}\mathbf{W}_e^T \tag{27}$$

Exploiting the skew-symmetry of $\mathbf{W}_e$ and setting $\boldsymbol{\lambda} = diag(\boldsymbol{\Lambda})$, we can derive a simple update equation:

$$(\lambda_i - \lambda_j)\mathbf{W}_{ije} = \mathbf{B}^*_{ij} \tag{28}$$

where $\mathbf{B}^* = \mathbf{B}_e - diag(\mathbf{B}_e)$. We solve this equation for each $\mathbf{W}_{ije}$. Finally we can compute the approximate gradients as $\frac{\partial \bar{\mathbf{F}}}{\partial \omega_e} = \bar{\mathbf{F}}\mathbf{W}_e$. The matrix $\mathbf{J}$ has a closed form derivative and we compute the derivative of $\mathbf{LL}^T$ using finite differences of the rigid body force samples, so $\frac{\partial \mathbf{JLL}^T\mathbf{J}^T}{\partial \omega_e}$ is straightforward to evaluate.

Appendix B: Enhanced Evaluation for $\mathbf{W}_e$

To evaluate $\mathbf{W}_e$, first we need to evaluate $\mathbf{B}_e$, and then compute $\mathbf{W}_e$ according to equations 27 and 28. Using equation 25 and expanding derivatives, we have:

$$\begin{aligned} &\mathbf{B}_e = \mathbf{B}_e^1 + \mathbf{B}_e^2 + (\mathbf{B}_e^1)^T \\ &\mathbf{B}_e^1 = \bar{\mathbf{F}}^T\frac{\partial \mathbf{J}}{\partial \omega_e}\mathbf{LL}^T\mathbf{J}^T\bar{\mathbf{F}} \qquad \mathbf{B}_e^2 = \bar{\mathbf{F}}^T\mathbf{J}\frac{\partial \mathbf{LL}^T}{\partial \omega_e}\mathbf{J}^T\bar{\mathbf{F}} \end{aligned} \tag{29}$$

First we consider the evaluation of $\mathbf{B}_e^1$. Because only the $\frac{\partial \mathbf{J}}{\partial \omega_e}$ factor is different from element to element, we can precompute $\mathbf{LL}^T\mathbf{J}^T\bar{\mathbf{F}} \in \mathbb{R}^{(n_u+6)\times r}$ in $O(n_u^2 r + mr)$ for all the elements. The major complication is to compute $\bar{\mathbf{F}}^T\frac{\partial \mathbf{J}}{\partial \omega_e}$ efficiently for all elements.

From Appendix C, $\frac{\partial \mathbf{J}}{\partial \omega_e}$ is composed of three parts:

$$\frac{\partial \mathbf{J}}{\partial \omega_e} = \begin{bmatrix} 0 & \hat{\mathbf{J}}^c & \hat{\mathbf{J}}^\tau \end{bmatrix} + \begin{bmatrix} 0 & \tilde{\mathbf{J}}_e^c & \tilde{\mathbf{J}}_e^r + \mathbf{J}^{com}\tilde{\mathbf{J}}_e^\tau \end{bmatrix} \tag{30}$$

where $\hat{\mathbf{J}}^c, \hat{\mathbf{J}}^\tau \in \mathbb{R}^{3n\times 3}$ is constant for all the elements, $\tilde{\mathbf{J}}_e^c$, $\tilde{\mathbf{J}}_e^r \in \mathbb{R}^{3n\times 3}$ is different from element to element (but its sparsity is of $O(c)$, where c is a constant), $\mathbf{J}^{com} \in \mathbb{R}^{3n\times 3}$ is a constant matrix, and $\tilde{\mathbf{J}}_e^\tau \in \mathbb{R}^{3\times 3}$ differs from element to element (but it is a small matrix).

Computing $\bar{\mathbf{F}}^T \frac{\partial \mathbf{J}}{\partial \omega_e}$ naively for all elements poses a complexity of $O(m^2 r)$. However,

$$\bar{\mathbf{F}}^T \frac{\partial \mathbf{J}}{\partial \omega_e} = \begin{bmatrix} 0 & \bar{\mathbf{F}}^T\hat{\mathbf{J}}^c & \bar{\mathbf{F}}^T\hat{\mathbf{J}}^\tau \end{bmatrix} + \begin{bmatrix} 0 & \bar{\mathbf{F}}^T\tilde{\mathbf{J}}_e^c & \bar{\mathbf{F}}^T\tilde{\mathbf{J}}_e^r + \bar{\mathbf{F}}^T\mathbf{J}^{com}\tilde{\mathbf{J}}_e^\tau \end{bmatrix} \tag{31}$$

where $\bar{\mathbf{F}}^T\hat{\mathbf{J}}^c, \bar{\mathbf{F}}^T\hat{\mathbf{J}}^\tau$ and $\mathbf{X} = \bar{\mathbf{F}}^T\mathbf{J}^{com}$ can be precomputed for all elements in $O(mr)$. The only terms which are needed per-element are $\bar{\mathbf{F}}^T\tilde{\mathbf{J}}_e^c, \bar{\mathbf{F}}^T\tilde{\mathbf{J}}_e^r$ and $\mathbf{X}\tilde{\mathbf{J}}_e^\tau$. Because the three matrices $\tilde{\mathbf{J}}_e^c, \tilde{\mathbf{J}}_e^r$ and $\tilde{\mathbf{J}}_e^\tau$ have a constant number of non-zero entries, this can be done for all elements in $O(mr)$. Therefore, computing $\bar{\mathbf{F}}^T \frac{\partial \mathbf{J}}{\partial \omega_e}$ can be reduced from $O(m^2 r)$ to $O(mr)$. The final matrix products $\bar{\mathbf{F}}^T \frac{\partial \mathbf{J}}{\partial \omega^e}$ and $(\mathbf{L}\mathbf{L}^T\mathbf{J}^T\bar{\mathbf{F}})$ take $O(mr^2)$ for all elements. Therefore, $\mathbf{B}_e^1$ can be computed in $O(mr^2)$.

Now consider the evaluation of $\mathbf{B}_e^2$. Finite differences are used to evaluate $\frac{\partial(\mathbf{L}\mathbf{L}^T)}{\partial \omega_e}$.

$$\frac{\partial(\mathbf{L}\mathbf{L}^T)}{\partial \omega_e} = \sum_{i=1}^{i=10} \frac{\partial(\mathbf{L}\mathbf{L}^T)}{\partial M_i}\frac{\partial M_i}{\partial \omega_e} \tag{32}$$

$\frac{\partial(\mathbf{L}\mathbf{L}^T)}{\partial M_i}$ are 10 matrices which can be precomputed in $O(n_u n_s)$ for all elements. The matrix product $(\bar{\mathbf{F}}^T\mathbf{J})\frac{\partial \mathbf{L}\mathbf{L}^T}{\partial \omega^e}(\mathbf{J}^T\bar{\mathbf{F}})$ takes $O(m(rn_u^2 + r^2 n_u))$ for all elements. This is the total complexity for $\mathbf{B}_e^2$.

This part can also be accelerated through further precomputation. Denote $\mathbf{A}_i = \frac{\partial(\mathbf{F}^T\mathbf{F})}{\partial M_i}$, $\mathbf{H} = \bar{\mathbf{F}}^T\mathbf{J} \in \mathbb{R}^{r\times(n_u+6)}$. We have:

$$\mathbf{B}_e^2 = \mathbf{H}\sum_{i=1}^{i=10}\mathbf{A}_i\frac{\partial M_i}{\partial \omega_e}\mathbf{H}^T = \sum_{i=1}^{i=10}\mathbf{H}\mathbf{A}_i\mathbf{H}^T\frac{\partial M_i}{\partial \omega_e} \tag{33}$$

Since only $\frac{\partial M_i}{\partial \omega_e}$ changes between elements, we can precompute $\mathbf{H}\mathbf{A}_i\mathbf{H}^T \in \mathbb{R}^{r\times r}$ before doing the summation. This reduces the complexity for evaluation of $\mathbf{B}_e^2$ to $O(mr^2)$, and also reduces the total complexity for $\mathbf{B}_e$ to $O(mr^2)$.

Appendix C: Evaluation of Derivative $\frac{\partial \mathbf{J}}{\partial \omega_e}$

Here we explain how to compute the derivative $\frac{\partial \mathbf{J}}{\partial \omega_e}$. The general form of $\mathbf{J}$ is shown in equation 24. The matrix $\mathbf{J}^m$ maps a surface index to full volume index. Because the FEM mesh does not change during optimization, $\frac{\partial \mathbf{J}^m}{\partial \omega_e} = 0$. We only need to consider last 6 columns of $\mathbf{J}$: $\begin{bmatrix} \mathbf{J}^{com} & \mathbf{J}^\tau \end{bmatrix}$.

Before going into the details of $\mathbf{J}^{com}$ and $\mathbf{J}^\tau$, denote the total mass of the object as M, and the total number of element nodes as n. The mass for each node i is m_i, the fill ratio for each element e is ω_e, and the mass of each element is $d_e = \rho\omega_e$, where ρ is the density.

Because we use hexahedral elements in the FEM mesh, we compute the node mass as:

$$m_i = \sum_{j\in N(i)} \frac{1}{8} d_j \tag{34}$$

where $N(i)$ denotes the set of element indices adjacent to node i.

The matrix $\mathbf{J}^{com}$ maps the center of mass force to each element. It is a tiled diagonal matrix of 3×3 of the following form:

$$\mathbf{J}^{com} = \frac{1}{M}\begin{bmatrix} \vdots & \vdots & \vdots \\ m_i & 0 & 0 \\ 0 & m_i & 0 \\ 0 & 0 & m_i \\ \vdots & \vdots & \vdots \end{bmatrix} \tag{35}$$

To compute $\frac{\partial \mathbf{J}^{com}}{\partial \omega_e}$ for element e, we need to evaluate:

$$\frac{\partial(m_i/M)}{\partial \omega_e} = \frac{\partial m_i}{\partial \omega_e}\frac{1}{M} - \frac{\partial M}{\partial \omega_e}\frac{m_i}{M^2}, \quad i = 1,\dots,n \tag{36}$$

The second term on the right hand side of equation 36 is non zero for all $i = 1,\dots,n$, because all element nodes contribute to the total mass M.

$$-\frac{\partial M}{\partial \omega_e}\frac{m_i}{M^2} = -\sum_{k\in N(e)}\frac{\partial m_k}{\partial \omega_e}\frac{m_i}{M^2} \tag{37}$$

Because element e contributes $\frac{1}{8}d_e$ to node m_k, we have

$$-\frac{1}{8}\sum_{k\in N(e)}\frac{\partial d_e}{\partial \omega_e}\frac{m_i}{M^2} = -\frac{1}{8}\sum_{k\in N(e)}\rho\frac{m_i}{M^2} = -\rho\frac{m_i}{M^2}, \quad i = 1,\dots,n \tag{38}$$

Denote this part of the derivative as $\hat{\mathbf{J}}^c$. It is a dense tiled matrix of the form:

$$\hat{\mathbf{J}}^c = \frac{\rho}{M^2}\begin{bmatrix} \vdots & \vdots & \vdots \\ -m_i & 0 & 0 \\ 0 & -m_i & 0 \\ 0 & 0 & -m_i \\ \vdots & \vdots & \vdots \end{bmatrix} \tag{39}$$

where every $i = 1\dots n$ is tiled with the 3×3 diagonal matrix.

Now consider the first part of equation 36, which is non zero only for nodes i that are adjacent to element e. We have:

$$\frac{\partial m_i}{\partial \omega_e}\frac{1}{M} = \frac{\partial m_k}{\partial \omega_e}\frac{1}{M} = \frac{\rho}{8M}, \quad for\ k\in N(e) \tag{40}$$

This part of the derivative is $\tilde{\mathbf{J}}_e^c$, a *sparse* matrix of the form:

$$\tilde{\mathbf{J}}_e^c = \begin{bmatrix} \vdots & \vdots & \vdots \\ \frac{\rho}{8M} & 0 & 0 \\ 0 & \frac{\rho}{8M} & 0 \\ 0 & 0 & \frac{\rho}{8M} \\ \vdots & \vdots & \vdots \end{bmatrix} \tag{41}$$

where only node indices $k \in N(e)$ are filled with the 3×3 diagonal matrix. So in summary:

$$\frac{\partial \mathbf{J}^{com}}{\partial \omega_e} = \hat{\mathbf{J}}^c + \tilde{\mathbf{J}}^c_e \tag{42}$$

The matrix $\mathbf{J}^\tau$ maps the center of mass torque to the elements. It is a tiled 3×3 skew-symmetric matrix, where each matrix denotes a cross product:

$$\mathbf{J}^\tau = \frac{1}{M}\begin{bmatrix} \vdots & \vdots & \vdots \\ 0 & \mathbf{r}_{iz}m_i & -\mathbf{r}_{iy}m_i \\ -\mathbf{r}_{iz}m_i & 0 & \mathbf{r}_{ix}m_i \\ \mathbf{r}_{iy}m_i & -\mathbf{r}_{ix}m_i & 0 \\ \vdots & \vdots & \vdots \end{bmatrix} \tag{43}$$

where $\mathbf{r}_i = \mathbf{p}_i - \mathbf{q}$ denotes the vector pointing from the center of mass ($\mathbf{q}$) to the position of node i ($\mathbf{p}_i$). Subscripts x, y, z denote the respective components of that vector. To compute its derivative, consider the z component:

$$\frac{\partial(\mathbf{r}_{iz}m_i/M)}{\partial\omega_e} = \frac{\partial\mathbf{r}_{iz}}{\partial\omega_e}\frac{m_i}{M} + \mathbf{r}_{iz}\frac{\partial(m_i/M)}{\partial\omega_e}, \quad i = 1,\ldots,N \tag{44}$$

The second term of 44 is computed with equations 36 and 40, and results in a constant part which is included in equation 49, as well as the following non-constant sparse term:

$$\tilde{\mathbf{J}}^r_e = \frac{\rho}{8M}\begin{bmatrix} \vdots & \vdots & \vdots \\ 0 & \mathbf{r}_{kz} & -\mathbf{r}_{ky} \\ -\mathbf{r}_{kz} & 0 & \mathbf{r}_{kx} \\ \mathbf{r}_{ky} & -\mathbf{r}_{kx} & 0 \\ \vdots & \vdots & \vdots \end{bmatrix}. \tag{45}$$

Similar to equation 41, only node indices $k \in N(e)$ are filled. For the first term in equation 44, we have:

$$\frac{\partial\mathbf{r}_{iz}}{\partial\omega_e}\frac{m_i}{M} = \frac{\partial(\mathbf{p}_{iz} - \mathbf{q}_z)}{\partial\omega_e}\frac{m_i}{M} = -\frac{\partial\mathbf{q}_z}{\partial\omega_e}\frac{m_i}{M} \tag{46}$$

Since $\mathbf{q}_z = \sum_{k=1}^{n}\mathbf{p}_{kz}m_k/M$, denote $\boldsymbol{s} = \sum_{k=1}^{N}\mathbf{p}_{kz}m_k$, we have:

$$-\frac{\partial(\boldsymbol{s}/M)}{\partial\omega_e}\frac{m_i}{M} = -\frac{\partial\boldsymbol{s}}{\partial\omega_e}\frac{m_i}{M^2} + \frac{\mathbf{q}_z m_i}{M^2}\frac{\partial M}{\partial\omega_e}, \quad i = 1,\ldots,N \tag{47}$$

The second part of this equation is equal to $\rho\frac{\mathbf{q}_z m_i}{M^2}$ for all i, so we have the total contribution to the constant part of derivative as:

$$(\mathbf{q}_z - \mathbf{r}_{iz})\frac{\rho m_i}{M^2}, \quad i = 1,\ldots,N \tag{48}$$

we denote this part as $\hat{\mathbf{J}}^\tau$, which is constant for all elements.

$$\hat{\mathbf{J}}^\tau = \frac{\rho}{M^2}\begin{bmatrix} \vdots & \vdots & \vdots \\ 0 & (\mathbf{q}_z - \mathbf{r}_{iz})m_i & -(\mathbf{q}_y - \mathbf{r}_{iy})m_i \\ -(\mathbf{q}_z - \mathbf{r}_{iz})m_i & 0 & (\mathbf{q}_x - \mathbf{r}_{ix})m_i \\ (\mathbf{q}_y - \mathbf{r}_{iy})m_i & -(\mathbf{q}_x - \mathbf{r}_{ix})m_i & 0 \\ \vdots & \vdots & \vdots \end{bmatrix} \tag{49}$$

For the first term of equation 47:

$$\begin{aligned} \frac{\partial\boldsymbol{s}}{\partial\omega_e}\frac{m_i}{M^2} &= -\sum_{k\in N(e)}\mathbf{p}_{kz}\frac{\partial m_k}{\partial\omega_e}\frac{m_i}{M^2} \\ &= -\sum_{k\in N(e)}\frac{1}{8}\mathbf{p}_{kz}\rho\frac{m_i}{M^2}, \quad i = 1,\ldots,N \end{aligned} \tag{50}$$

The summation of node position $\sum_{k\in N(e)}\frac{1}{8}\mathbf{p}_{kz}$ is the z component of the position of element e. Using the z component of the centroid of element e: $\mathbf{t}_{ez} = \sum_{k\in N(e)}\frac{1}{8}\mathbf{p}_{kz}$, equation 50 can be computed as:

$$\frac{1}{M}\begin{bmatrix} \vdots & \vdots & \vdots \\ m_i & 0 & 0 \\ 0 & m_i & 0 \\ 0 & 0 & m_i \\ \vdots & \vdots & \vdots \end{bmatrix}\frac{\rho}{M}\begin{bmatrix} 0 & -\mathbf{t}_{ze} & \mathbf{t}_{ye} \\ \mathbf{t}_{ze} & 0 & -\mathbf{t}_{xe} \\ -\mathbf{t}_{ye} & \mathbf{t}_{xe} & 0 \end{bmatrix} = \mathbf{J}^{com}\tilde{\mathbf{J}}^\tau_e \tag{51}$$

where the matrix $\tilde{\mathbf{J}}^\tau_e$ denotes the right 3×3 matrix of element e. In summary, the derivative of $\mathbf{J}$ is:

$$\frac{\partial\mathbf{J}}{\partial\omega_e} = \begin{bmatrix} 0 & \hat{\mathbf{J}}^c & \hat{\mathbf{J}}^\tau \end{bmatrix} + \begin{bmatrix} 0 & \tilde{\mathbf{J}}^c_e & \tilde{\mathbf{J}}^r_e + \mathbf{J}^{com}\tilde{\mathbf{J}}^\tau_e \end{bmatrix} \tag{52}$$

Appendix D: Optimization with different SIMP parameters

SIMP penalizes intermediate densities in structural optimization by exponentiating them when computing the stiffness matrix. This makes the intermediate densities uneconomical in the optimization, attempting to force them to go to either zero or one. The exponent is a parameter, where increasing it will further penalize the intermediate densities, but render the optimization more difficult.

In figure 5, we use SIMP with an exponent of 5. We experimented with different choices and found it does not improve the results. As shown in figure 17, small SIMP exponents (2 or 3) lead to an unacceptable number of intermediate densities. Increasing the exponent above 10 often stall the optimization at the very beginning. We need a different approach to overcome the local minima.

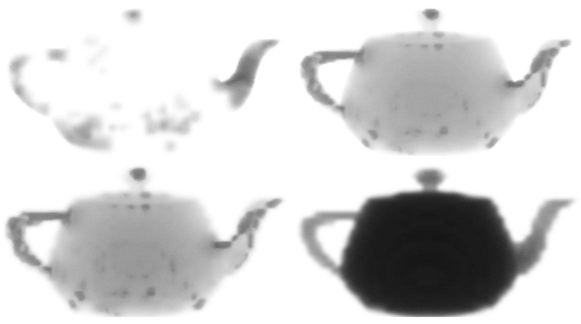


Figure 17: *Top row: Converged results using SIMP exponent 2,3 respectively. Bottom row: Converged results using SIMP exponent 5, 10, respectively.*